

WL-TR-94-8032

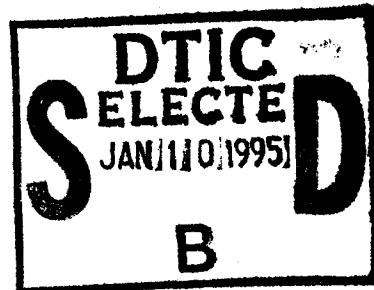


**NEXT GENERATION CONTROLLER
SPECIFICATION FOR AN OPEN SYSTEMS ARCHITECTURE
STANDARD - OVERVIEW**

Martin Marietta Astronautics
PO Box 179
Denver, CO 80201

28 September 1994

Final Report For the Period March 1989 - August 1994



Approved for Public Release; Distribution is Unlimited.

DTIC QUALITY INSPECTED 3

Manufacturing Technology Directorate
Wright Laboratory
Air Force Materiel Command
Wright-Patterson Air Force Base, Ohio 45433-7739

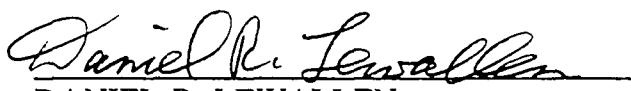
19950106 083

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASC/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.



Daniel R. Lewallen
DANIEL R. LEWALLEN
Project Manager



Michael F. Hitchcock
MICHAEL F. HITCHCOCK
Supervisor



Bruce A. Rasmussen
BRUCE A. RASMUSSEN, Chief
Integration Technology Division
Manufacturing Technology Directorate

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WL/MTIA, Bldg. 653, 2977 P St., Suite 6, W-PAFB, OH 45433-7739 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB NO. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED	
	28 September 1994	Final March 1989 - August 1994	
4. TITLE AND SUBTITLE		5. FUNDING NUMBERS	
Next Generation Controller Specification for an Open Systems Architecture Standard - Overview		C F33615-89-C-5706 PE 78011 PR 3095 TA 06 WU 18	
6. AUTHOR(S)		8. PERFORMING ORGANIZATION REPORT NUMBER	
Martin Marietta Astronautics PO Box 179 Denver, CO 80201			
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REP NUMBER	
Manufacturing Technology Directorate Wright Laboratory Air Force Materiel Command Wright-Patterson AFB, OH 45433-7739		WL-TR-94-8032	
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION/AVAILABILITY STATEMENT		12b. DISTRIBUTION CODE	
Approved for Public Release; Distribution is Unlimited.			
13. ABSTRACT This report describes an open architecture framework for advanced manufacturing controllers including machine tools, robotics, process control, and coordinate measurement. The methodology described is applicable to both system design as well as system retrofit applications. A component-based approach is used that allows system architectures to be developed from generic building blocks that address system requirements and resulting functionality requirements. Initial architectures are used to produce an "application architecture" that addresses specific system messaging requirement. An "application system" is then synthesized using off-the-shelf, reusable "implementation components" for the application software. Hardware/software specific platform issues are addressed through the use of system "profiles," a convention adopted from the IEEE Portable Operating system Interface (POSIX) standard that is now gaining wide acceptance. A number of examples are provided demonstrating use of all aspects of the methodology on a typical machine tool controller application. Appendices provide details with respect to specific components, requirements, and domain models.			
14. SUBJECT TERMS		15. NUMBER OF PAGES	39
Open System Architecture, Computer Numerical Control (CNC), Machine Tool Control, Reusable Software, Hardware/Software Platform.		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASS OF THIS PAGE	19. SECURITY CLASS OF ABSTRACT	20. LIMITATION ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std Z239-18
298-102

FOREWORD

The purpose of this document is to provide a description of the Next Generation Workstation/Machine Controller (NGC) program philosophy and objectives in sufficient detail so to be able to quickly and efficiently assess the impact on emerging system designs and concepts. The focus for this document is the Specification for an Open System Architecture Standard (SOSAS) that is the primary deliverable of the NGC program. The SOSAS will capture definitions, conventions, and standards as they relate to the final NGC family of controllers. The document presented here is not meant to be construed as the final SOSAS, but rather it represents the key elements of the SOSAS necessary to achieve the desired objectives relating to open systems, interchangeability, interoperability, portability, etc. It is fully anticipated that subsequent discussion of this document will provide some of the most important feedback with respect to the production of the final SOSAS document. In this respect, comments relating to this document are strongly encouraged.

This is an NGC SOSAS overview. The complete document is also available entitled "Next Generation Controller Specification for an Open Systems Architecture Standard".

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution _____	
Availability Codes	
Dist	Avail and/or Special
A-1	

TABLE OF CONTENTS

1.0 INTRODUCTION.....	1
1.1 Technology Infusion.....	2
1.2 Maintenance.....	2
1.3 Operator Training	2
2.0 NGC SYSTEM PHILOSOPHY AND STRUCTURE.....	6
2.1 Introduction	6
2.2 Industrial Control Domains	8
2.3 NGC Planning and Execution in the Manufacturing Context	10
2.4 NGC Development Process	12
2.5 Structural Aspects of NGC.....	16
2.6 Open System Architecture Specification.....	20
2.7 Open System Environment Overview and Profiles.....	20
2.8 Implementation Issues	26
2.9 NGC Tools and Libraries	29
2.10 Conformance Overview.....	30
2.11 Growth and Evolution	31
3.0 SUMMARY AND RECOMMENDATIONS.....	32
3.1 Summary	32
3.2 Synergistic Activities.....	34
3.3 Recommendation	35

LIST OF FIGURES

Figure 1	Manufacturing Enterprise Hierarchy	7
Figure 2	Integration Architecture	8
Figure 3.	Domains of Practice for Industrial Control.....	9
Figure 4a	Traditional Flow of NC Machine Tool Programming.....	10
Figure 4b.	NGC Concept of Product Design and Production Planning.....	12
Figure 5a	NGC Development Process	13
Figure 5b	From Reference Requirements to Implementation.....	14
Figure 6	NGC Architecture Perspective.....	16
Figure 7	NGC Realization Process	17
Figure 8	NGC Timing Domains	19
Figure 9	NGC OSE Reference Model	21
Figure 10	NGC OSE Taxonomy	22
Figure 11	Communications Standards Architectural Hierarchy	23
Figure 12	NGC EEI Interoperability Standards—Physical Mapping	24
Figure 13a	API Profile Suite.....	25
Figure 13b	NGC Profiles—Template Structure.....	26
Figure 14	Example of an Application Architecture for a Machining Center	27
Figure 15a	Platform Partitioning, Example 1	28
Figure 15b	Platform Partitioning, Example 2	28
Figure 16	Platform Implementation Examples	29
Figure 17	Implementation Component Library & Tools—Structure & Contents.....	30
Figure 18	NGC Document Vision	32

1.0 INTRODUCTION

The Specification for an Open System Architectural Standard (SOSAS) is a proposed specification to govern the design and construction of a family of workstation/machine controllers. The goals of the SOSAS are to define an open systems architecture that enables significantly increased capability, flexibility, and improved price-to-performance ratios for workstation/machine controllers. Applications of this architectural specification include machine tools, robots, process control, coordinate measurement machines, and similar applications.

The Next Generation Workstation/Machine Controller (NGC) program which has produced the SOSAS is a result of a directive from President Ronald Reagan in 1987 to urge U.S. industry to "forge a broad, working partnership...to out-compete foreign nations." This Presidential technology program was developed with input from the National Center for Manufacturing Sciences (NCMS) and was awarded by the Air Force Manufacturing Technology Directorate in the fall of 1989.

The NGC program philosophy of systems that are interoperable, interchangeable, and portable is an outgrowth of a more general trend in all areas of systems development that stress "openness." Like many new ideas in engineering, the concept of an open system is one that has been much easier to describe qualitatively than it has been to establish rigorous design methodologies. In the arena of advanced manufacturing control systems, NGC and the SOSAS are an attempt to merge the evolving methodology of open systems with a well-established technology base in machine tool, robotics, measurement, and process control. As in any emerging technology, growth and evolution do not come without some degree of controversy and argument. There is no lack of recognition on the part of the NGC development team that while the long-term benefits of an open approach to manufacturing controller design are obvious to virtually everyone, the burden of making this vision a reality will fall on a vendor community that is engaged in an intensive economic struggle and is faced, on a daily basis, with the necessity of looking at an unforgiving bottom line. The result is that the NGC development process has never been viewed as a "do it all over again" exercise, i.e., starts with a clean sheet of paper. Instead, it has been viewed as "given the space of open system solutions, pick the solution that is a closest fit to existing systems."

The NGC can be viewed from many different perspectives. While the control builder will ultimately be the one tasked with turning ideas and paper into working hardware and software, it is the end user of manufacturing control systems who will, in the long term, dictate the final form and structure of these evolving open systems. It is the end user who will apply the resulting systems in an attempt to produce products that are cheaper than the competitor and as a result, the end users that are successful in this process will, explicitly or implicitly, determine the marketplace forces that ultimately shape the technology. There are many complexities involved in defining what leads to a cheaper product from the standpoint of the end user. While the actual dollar cost of a manufacturing controller will always remain an important aspect of the overall metric, there are many other factors that influence the end user's manufacturing strategy.

1.1 Technology Infusion

The ability to rapidly integrate new technologies and practices has always been, and will always remain, a crucial factor in competitiveness. A new technology can appear in many different ways and impact practice, hardware, software, or all of these. A good example of this is the emerging trend towards machine tool controllers that makes use of temperature and/or vibration data at the spindle. This requires not only the ability to integrate new sensors into the system but also the ability to rapidly and efficiently modify existing software structures in order to get the right data to the right place in an appropriately timely fashion and implement newer and potentially more complex control laws that take advantage of the new data.

1.2 Maintenance

Key factors in the competitiveness of any system are the supporting elements necessary to keep the system running. In general, commonality of system elements leads to lower system cost for a variety of reasons. These include a decrease in inventory size for spare components, the standardization of procedures for diagnosing and repairing systems, and the ability to use trained diagnostic and repair personnel across a wider variety of systems because of the decrease in specialization.

1.3 Operator Training

Commonality of systems decreases the amount of additional training that is required to transition operators from one system to another. Major retraining can be supplanted by incremental training based upon knowledge of a core system structure. In this case, an

open system architecture that utilizes standardized system features can achieve nearly the same desired commonality as a mandate for all systems from one manufacturer. From the standpoint of the operator, common look and feel of interfaces is more important than the internal system that generates the look and feel.

While the issue of the specific attributes desired in a NGC controller were derived from both the *Needs Analysis* document and the *Requirements Definition Document*, it is obvious that the resulting NGC systems must help the end user in a number of ways; neither very high-technology nor extremely low-cost systems are in themselves the full answer.

Instead of attempting to mandate a specific point solution to the advanced manufacturing controller problem, a philosophy is embodied herein that leads to maximum flexibility with the respect to ability to modify both the hardware and the software in the system to achieve desired enhancements in capability as necessary.

To facilitate a standardized approach to developing the structural elements of a system, a component based approach has been adopted as a basic building block for the SOSAS. The fundamental attributes of this approach are described including the use of a *reference architecture* that is comprised of *primitive components* that are used to delineate system requirements and structure in terms of generic "building blocks". From the reference architecture an *application architecture* is constructed that captures the functionality of the end system at an abstract level. While a level of abstraction above the final hardware and software system, the application architecture allows responsibilities and dependencies to be clearly established. Finally, through the selection of *implementation components* a final software structure is determined. The step from application architecture to implementation is a very difficult one because it involves the specifics of the system platform. The platform includes all system hardware as well other system software such as operating systems, communication software, etc.

Effectively dealing with platform issues requires dealing with the specifics of different types of potential hardware solutions as well as a large number of standards and conventions that have arisen with respect to communication, operating systems, device interfaces, graphics representations, etc. Because NGC has the goal of establishing a foundation for the introduction of open systems technology into the advanced manufacturing arena, it was felt that it would be a mistake for a small group to choose a relatively small set of possible standards and conventions that would then exclusively serve the NGC community. As a

result, it was necessary to find a means of accommodating systems that could draw on a wide range of hardware and software solutions in arriving at a system implementation. This problem was dealt with through the use of a representation concept known as a *profile*. This concept is crucial to understanding how the NGC specification can achieve the open system objectives without overly constraining system designers. Profiles can be thought of as a means of classifying NGC-compliant systems. The concept of the profile was adopted from POSIX (Portable Operating System Interface) literature. (POSIX and its role in NGC are discussed in greater depth later in this document.) The vendor will use profiles to succinctly and unambiguously describe the system structure. With a profile specified for a specific system, the end user will be able to determine whether or not the candidate system has the flexibility necessary for a given set of applications. A very simplified type of profile is now commonly used to describe software packages as being "Mac" or "PC" compliant. Similarly, the designation of "DOS" or "Windows" can be thought of as a subprofile under PC. This flexibility assessment could include the ability to operate with existing software packages, the ability to communicate with other factory systems with minor modifications, or the ability to expand hardware features such as input/output (I/O) or motor drivers. NGC does not preclude a vendor from building a system that is essentially non-open; it does, however, guarantee that a *system purchaser can be assured of understanding precisely what degree of openness is being provided*. Necessarily, standards play an important role in achieving the NGC goals.

The primary issues that impact the further evolution of NGC are the availability of libraries of NGC components (both primitive and implementation) that can be made widely available to the community at large for incorporation into emerging systems and the development and availability of tools that will facilitate the NGC development process. While an initial set of primitive components are provided for machining applications as an appendix to the full report, sufficient raw material for a robust initial implementation component library through the work of the National Institute of Standards and Technology (NIST) as an output of the Enhanced Machine Controller (EMC) program. Therefore, the key issue is the establishment of a repository for the initial library.

The issue of tools is more complex but not intractable. It is generally believed that the tool technology necessary for NGC currently exists through a variety of existing developmental and commercial packages. Ideally, a complete tool set could be integrated and made available to the general community. This would be the fastest way to spur the growth of NGC type systems. Unfortunately, the resources for such an activity have not yet been

identified. The ARPA Domain Specific Software Architecture (DSSA) program is in the process of defining, developing, and validating many tools that would directly fulfill requirements identified for NGC. Therefore, a programmatic relationship with the DSSA activity is one possible method of establishing an initial NGC tool set.

2.0 NGC SYSTEM PHILOSOPHY AND STRUCTURE

2.1 Introduction

The open system concepts specified in this document establish a framework for the design and construction of a family of workstation/machine controllers for industrial machines. This framework addresses issues associated with both application software and the hardware software platforms on which this application software will run. This NGC specification is based where possible on *de jure* and *de facto* open standards in manufacturing, controls, and computing technology but provides a sufficiently robust structure such that evolving and new standards and technology can readily be incorporated. The NGC specification facilitates commonality of components across a complete range of machine controllers. These controllers are to be used in a wide variety of manufacturing operations that include machining, robotics, and inspection.

A NGC supports a wide range of processing and discrete part manufacturing applications, including machine tools of all types, robots, electronic assembly, material handling devices, inspection devices, and virtually all types of automated equipment in both manned and unmanned environments and networked and stand-alone configurations. Specifically, a NGC controls a manufacturing workstation, which is defined as a single material transformer and related material handling and inspection equipment. As shown in Figure 1, the manufacturing workstation is placed at the lowest level in the hierarchy of the overall manufacturing enterprise. Several manufacturing workstations, each controlled by an NGC, are managed by a cell. Multiple cells are managed by a center, and multiple centers are managed by a factory. At the top of the hierarchy, the enterprise manages multiple factories. While this specification focuses on the lowest level of the this hierarchy, the results and structure are immediately and easily extended to include the higher hierarchical levels on an evolutionary basis. Therefore, a foundation is established for much more complex Computer Integrated Manufacturing (CIM) systems that will retain the open systems structure described in this document.

The open systems concepts are evolving from requirements established by the NGC community of control builders, machine integrators, end users, members of standards organizations, and university researchers. It is flexible enough to cover the broad range of

manufacturing practice, and it is extendible to absorb future advances in technology while accommodating existing manufacturing and controller practice.

To produce an enduring standard, manufactured products and manufacturing processes must be described independently of specific equipment, methods, or topologies. An architecture, calling for a specific topological arrangement and interconnection of components, does not provide enough flexibility to control every application in discrete part manufacturing, nor does it accommodate technological advances over a period of decades. Moreover, several topologies may be required in order to develop a system and understand its operation under a variety of situations. The overall view of the system architecture that is used in the NGC process is captured in the *Integration Architecture*. (Figure 2).

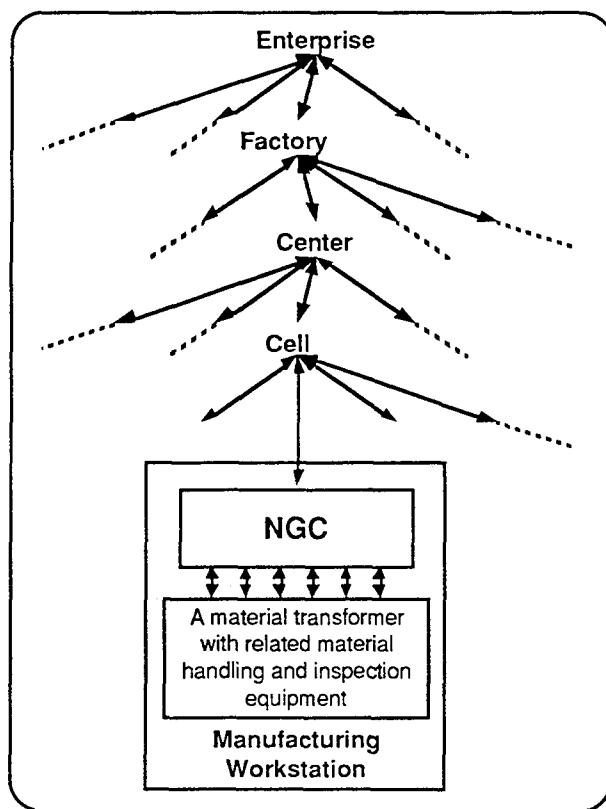


Figure 1 Manufacturing Enterprise Hierarchy

The integration architecture, Figure 2, combines manufacturing and control application components with a supporting infrastructure. It also acts as a framework for incorporating open and de facto standards. Applications components intercommunicate by messaging and can be adapted to a variety of component interconnection topologies. Typical

computing platform conventions, definitions, and capabilities are provided by the Open Systems Environment (OSE).

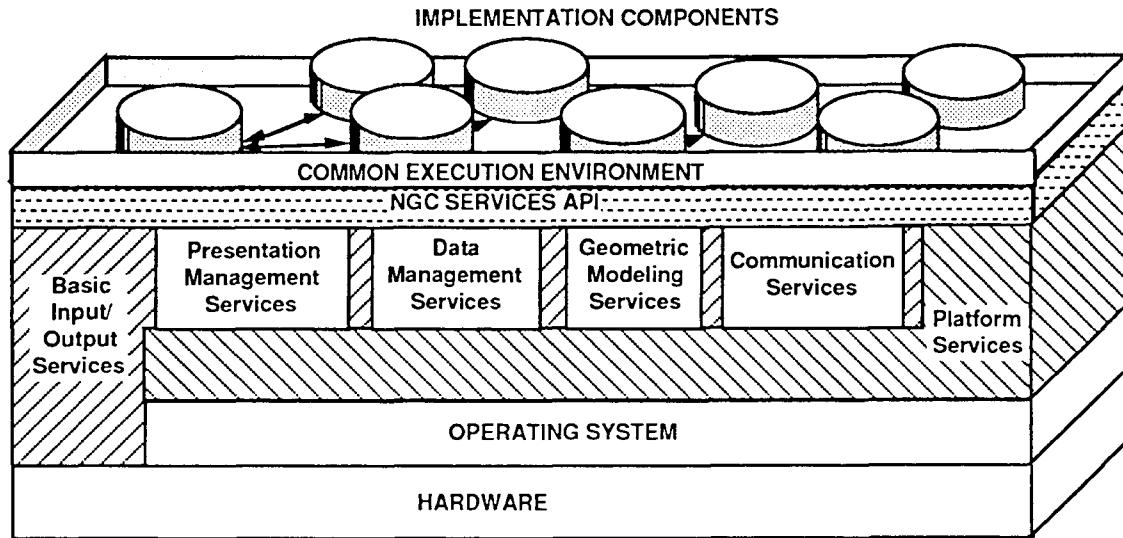


Figure 2 Integration Architecture

2.2 Industrial Control Domains

Manufacturing enterprises are complex, information-intensive environments. The practices associated with discrete part manufacturing can be grouped into three principal domains: (1) manufacturing practice, (2) controller practice, and (3) computing practice. The three domains and some associated concepts are shown in Figure 3. Terminology and representations from these domains are used in this specification. Existing and emerging standards, developed within the purview of these domains, are incorporated in this specification by reference. Thus the considerable investment and effort devoted to the three domains, especially computing practice, can be reused effectively to support manufacturing applications.

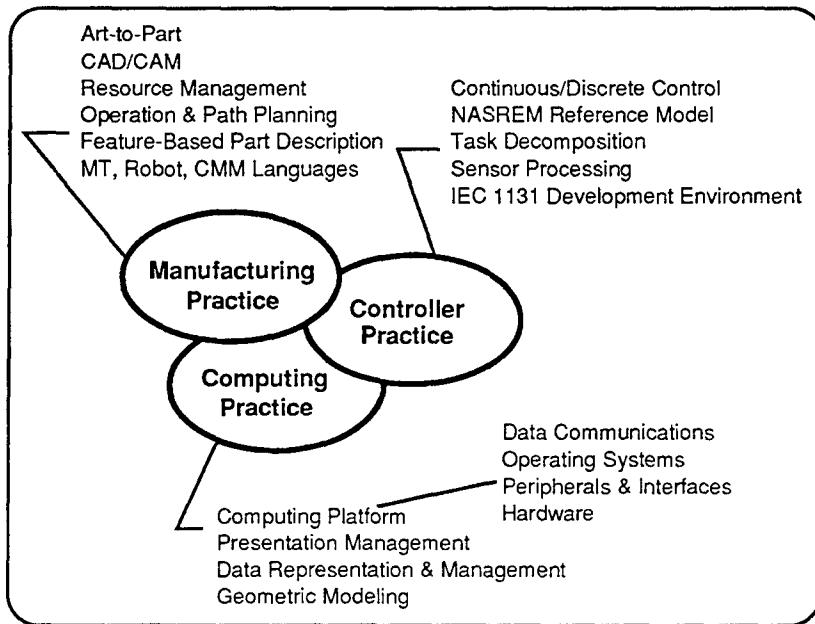


Figure 3. Domains of Practice for Industrial Control

Manufacturing practice covers the process of transforming raw materials into finished parts. The underlying concept for the discrete part manufacturing process is art-to-part. The process begins by capturing a part design using a computer-aided design (CAD) system. The resources for making the part are scheduled, and the machining, material handling, and measuring activities are planned using a computer-aided manufacturing (CAM) system. Motion paths are planned and used to drive the appropriate machinery, i.e., tool paths for machining, robot end-effector paths for manipulation, and probe paths for measurement. Some related manufacturing practice standards include those for feature-based part description, part programming, and measurement languages. Many of the steps in this process are done manually or disconnected. In the future, the art-to-part process is envisioned as a seamless information flow from the designer's concept to the finished part.

Controller practice covers the organization and control of manufacturing equipment. Equipment is controlled mostly by closed-loop controllers, which provide continuous (motion) control and discrete control.

Computing practice covers the computing and communication technologies required to support manufacturing and controller practice. The practice includes design and use of computer hardware and software. Computing practice is the target of intense open systems activity, which benefits NGC as it evolves and becomes sufficiently mature.

2.3 NGC Planning and Execution in the Manufacturing Context

The traditional manufacturing practice and general flow of numerically-controlled (NC) machine tool programming starts with a design review by a part programmer as shown in Figure 4a. The part programmer's role has been to analyze the product design and determine how to make the part economically, the machining range of workpiece, the method of mounting the workpiece on the machine tool, the machining sequence of every operation, and the cutting tools and cutting conditions. The output of the part programmer has been a part programming manuscript that documents the logical order of machine operations. The part geometry and cutter location data are post processed to a specific machine-readable format and transferred to the machine control unit (MCU).

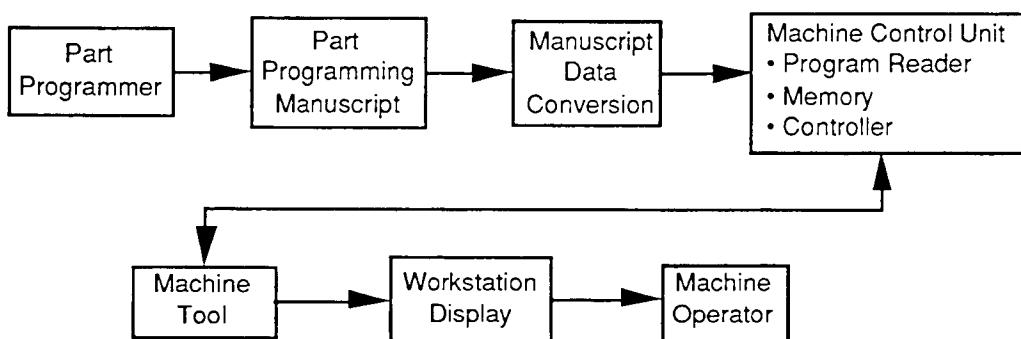


Figure 4a. Traditional Flow of NC Machine Tool Programming

The MCU memory has an executive program, program reader, shop floor programming language, cutter line control, machine interface logic, program subroutines or canned cycles, and tool changer control. The program reader in the MCU converts the coded instructions and the MCU controller generates an output signal to servo mechanisms to drive and direct the machine tool. The machine tool resolver or feedback device determines the precise location of the workpiece relative to the cutter and returns this data to the machine controller and also to machine operator workstation display. The workstation display may display full operational and parametric data; display job setup instructions; and have the capability for program verification, editing and update, maintenance and troubleshooting, fault messages, and graphical representations of the workpiece, tools, and cutter paths. The responsibility of the machine operator is to monitor the machine operation, monitor workpiece loading and unloading, provide information feedback, and perform operator programming through manual data input.

The intent of a NGC is to provide the product designers, process planners, NC programmers, machine operators, and factory managers with more flexibility in planning and committing factory resources. This increased flexibility and efficiency will be feasible through industrial machine controllers which allow open exchange of planning data, feature-based part representation, cutter location data, tooling and fixture data, and operations sequence data among different types machine tools without rewriting the part programs source code, and eliminating programming and processing for specific machines.

The concept of NGC product design and production planning is shown in Figure 4b. The product designer is able to generate a CAD that specifies the geometric features and considers material handling, assembly requirements, and manufacturability. Part geometry and features are interpreted through a CAM system to develop the process plan and generate the machining program. A computer-aided process planning (CAPP) system would evaluate economical production based upon a part family data base that has a library of part features, geometric data representations, and machining processing data. The process planning system determines the machine routing and operations sequence, methods, fixtures and tooling, and setups. From the process plan and machining program, a NC program is sent to a NGC workstation controller.

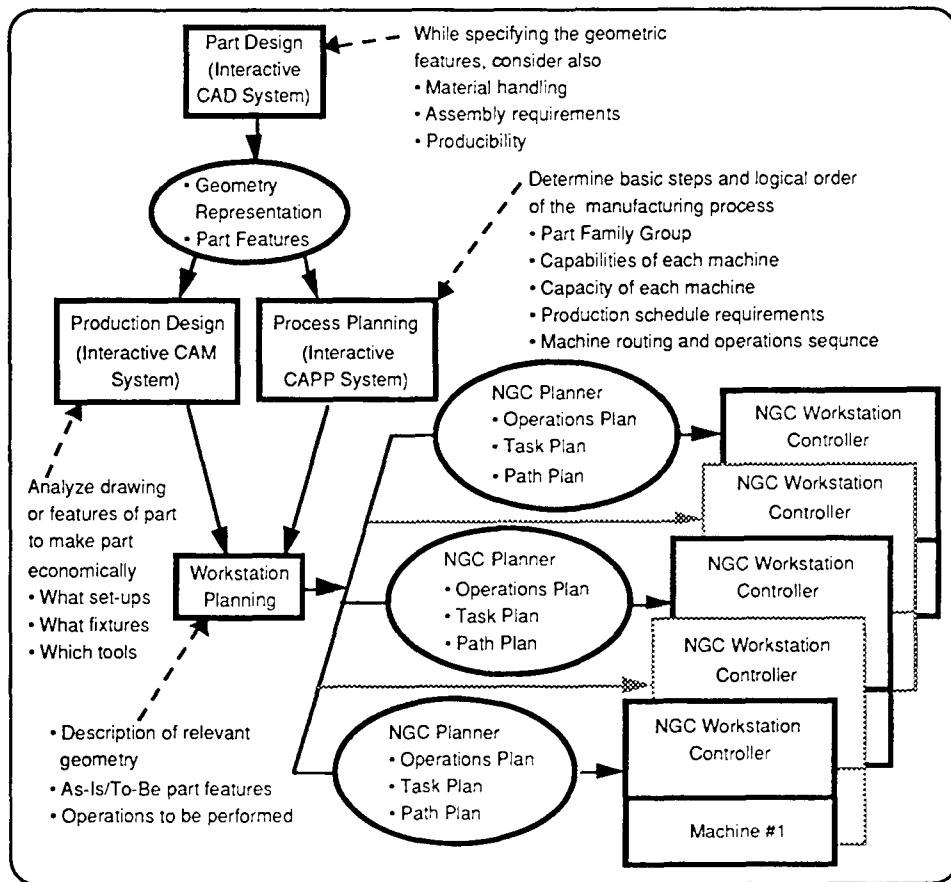


Figure 4b. NGC Concept of Product Design and Production Planning

2.4 NGC Development Process

To achieve the goals of the NGC program, it is necessary that the SOSAS structure adequately addresses all of the primary issues associated with system structure, application software, and platform development. Without a roadmap that shows how all of these elements are accommodated by the SOSAS, it is difficult to appreciate how the SOSAS supports all of these areas: (1) the bottoms-up development of a fully integrated system, (2) development and integration of new system software, and (3) expansion of platform capabilities to accommodate new requirements. A key element of the NGC philosophy is the ability to achieve rapid and effective system modifications that address only the part that "needs to be fixed" and do not necessarily entail massive, expensive system re-design.

A roadmap for the SOSAS development process is shown in Figures 5a and 5b, which illustrate the basic design and development paths for addressing system structural issues and platform issues. It is important to realize that any block on the diagram can be

considered to be an entry point for the design process. Development of new system application software, for example, would entail only the "upper" path with the platform definition being considered fixed. Similarly, if only the platform structure is being considered for modification, then the application architecture elements can be considered as given from the standpoint of platform issues and therefore, definition of the new platform profile is the key issue.

The key to understanding Figure 5a is the realization that system structure is embodied in the specification of an **application architecture**. The "upper" design path results in the application architecture, **reference requirements** are compared with **user needs** to derive a set of problem specific **application requirements**. By using a well-defined set of reference requirements as the basis for selecting the final system application requirements, a tremendous amount of consistency is brought to the entire requirements process, something not always achieved in ad hoc requirements derivation processes.

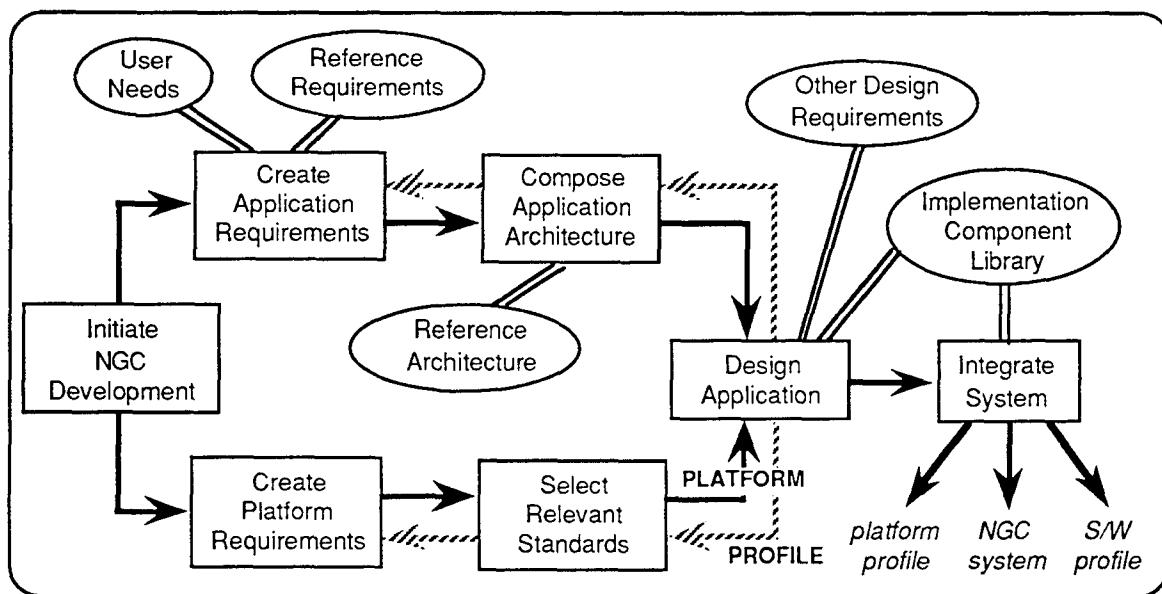


Figure 5-a NGC Development Process

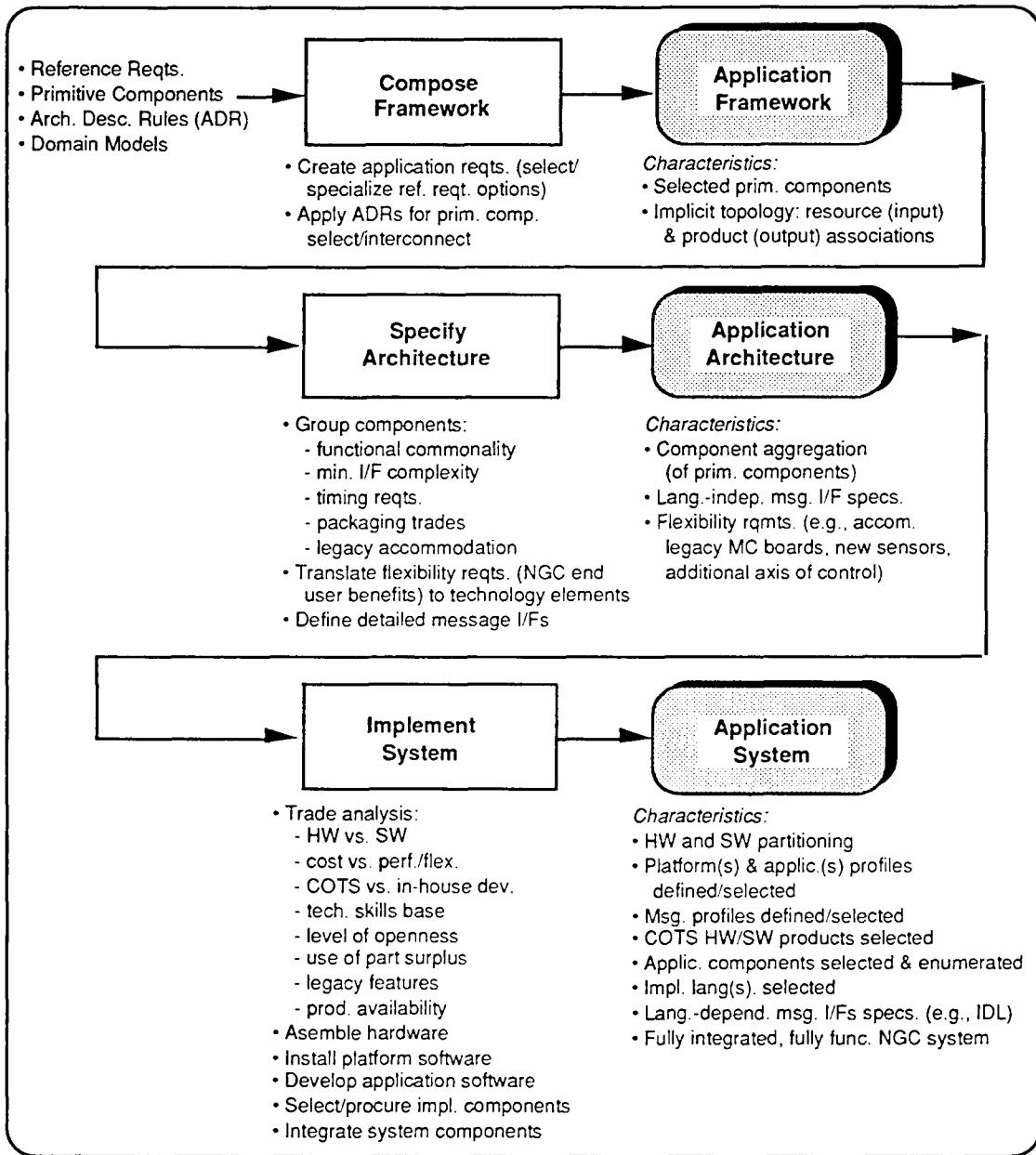


Figure 5b From Reference Requirements to Implementation

The **Reference Architecture** is the key to turning application requirements into the application architecture. The Reference Architecture consists of both **primitive** and **aggregate components**. Components are abstract building block elements that describe functionality and communication. The application architecture is built from these components. Although the Reference Architecture will necessarily be a "living" library that continually grows, the objective is to develop a set of components that "span the space" of desired functionality and communication for establishing a system structure. Therefore, in

the system design process, when the Reference Architecture is found insufficient, new components are added to the reference architecture. The application architecture is a complete and consistent topology for the representation of the system.

The lower pathway of Figure 5a addresses the issue of how the system will be physically implemented from the standpoint of processors, buses, communication, I/O, etc. It is not an objective of the SOSAS to enforce a particular design philosophy with respect to the system platform structure. The NGC does not attempt to lead all system designers in the direction of a standard "box". What it does do, however, is to establish a methodology for accurately capturing what has been produced by a specific vendor.

Platform requirements are used to select the standards (or conventions) that the designer feels are necessary for a specific platform implementation based on heritage, cost, performance, etc. Once the basic platform structure has been established, the key elements of the design are documented in the form of a **platform profile**. The notion of the platform profile was derived from the realization that (1) because of diverse market forces there will never be a "one platform does it all" to satisfy everyone, and (2) from the standpoint of achieving system openness with all of the associated "-ilities" it is of crucial importance that the user is able to understand what is being purchased in such a way that system expansion and extension can readily be assessed. The platform profile, by documenting the key standards and conventions used to construct the system, provides this insight.

The application architecture and platform design (as reflected in the profile) are unified in the **design application** phase of the process. Here, the **implementation component library** is used to take advantage of off-the-shelf software elements to instantiate the application architecture and insure consistency with the platform profile. Abstract functionality is replaced by actual software that includes both the functional aspects as well as the practical man-specific aspects necessary to accommodate the chosen platform profile.

Figure 6 is essentially a reiteration of the Integration Architecture shown earlier. However, this figure better illustrates how applications software, hardware platform elements, and typical COTS software products, e.g. operating systems, combine into an overall system that meets the NGC requirements for openness.

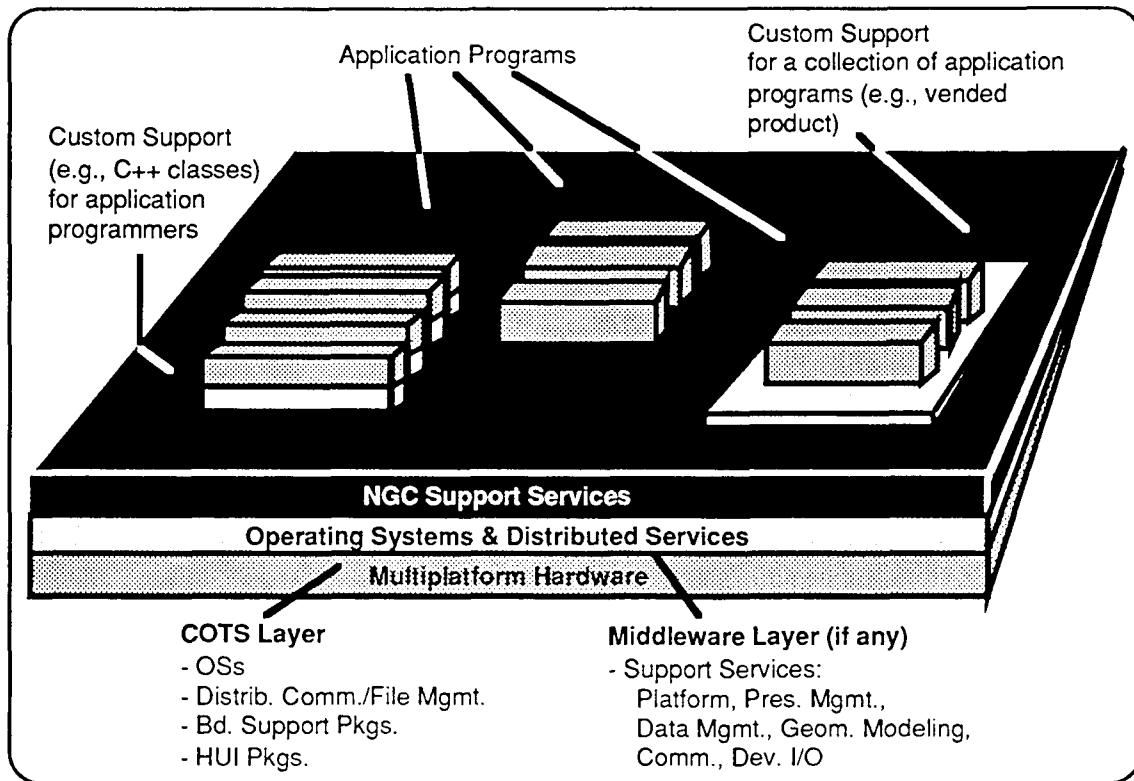


Figure 6 NGC Architecture Perspective

2.5 Structural Aspects of NGC

The NGC open system is comprised of application software exchanging information via data communication mechanisms and connected to the services provided by operating systems and hardware through a common interface called an application program interface (API). The software is implemented in the form of components that interchange messages in the process of carrying out their responsibilities. Components run under a Common Execution Environment (CEE) that provides for transparent peer-to-peer message exchange between the components as well as other services.

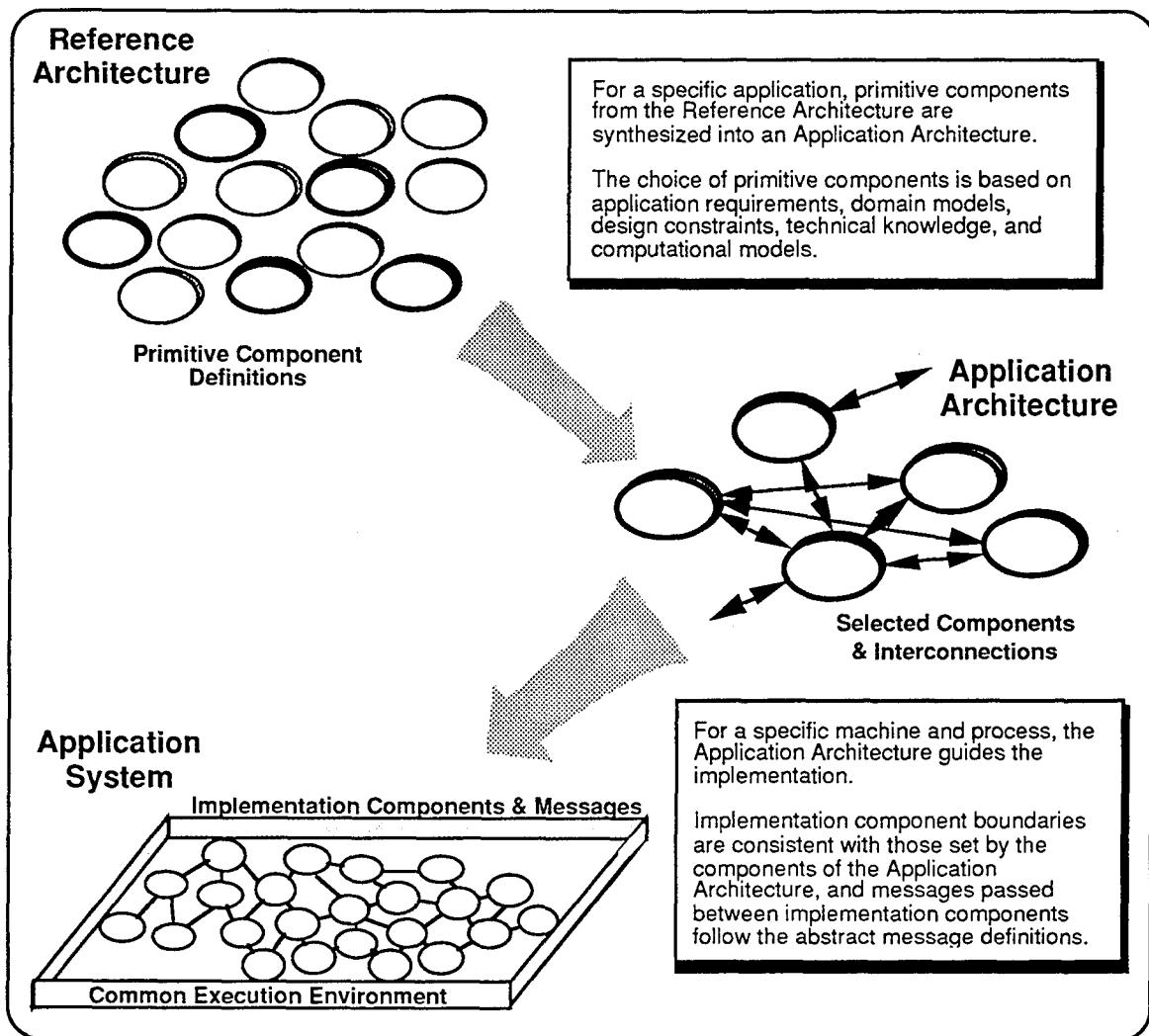


Figure 7 NGC Realization Process

Each component is a single separate thread of execution within the CEE since a component must be able to send and receive messages independently from the other components. A component encapsulates the data and functionality it needs to carry out its assigned responsibility. This approach is extremely flexible compared with older hard-wired systems, allowing the NGC system software to be reconfigured dynamically during operation by activating or deactivating agents. Dynamic reconfiguration does not cover the addition or removal of hardware; connecting or disconnecting hardware will require a system shutdown.

Components have the following attributes:

- Responsibility—The role the component plays, distinguishing it from the other components in the CEE, in the successful overall operation of the NGC system;
- Peer-to-peer Relationship(s)—The collaborative relationships the component has with other components as required to carry out its responsibility;
- Behavior(s)—The specific functionality encapsulated by the component, where each behavior is expressed as one or more operations to be performed in response to a message;
- Message(s)—The complete set of specific instructions necessary for evoking all of the behaviors encapsulated by the component. These messages must be defined in enough detail to guarantee interoperability.
- Application Program Interface(s)—The interface(s) a component uses specifically to access services provided by the Open Systems Environment (OSE).

For a specific controller application, a configuration is an information structure that provides a description of all required hardware and software elements and their interconnections. The configuration contains the information needed to maintain safe and reliable operation of the controller in carrying out all of its intended responsibilities. The configuration includes, as a minimum, the description of all of the required components. A directory service component or group of components has the responsibility for assuring that all of the components needed for reliable controller operation are available and working properly in the CEE. A configuration is used by the directory service component(s) to maintain a roster of active components and facilitate message interchange among them. The roster of active components will change dynamically, for example, with a shift in the operational requirements of the controller as it carries out its various responsibilities.

The NGC spans three separate timing domains: non real time (non RT), real time (RT), and hard real time (hard RT). This partitioning of the timing requirements is shown in Figure 8. The non-RT domain of standard computing covers activities like compiling and linking software, reading data from a file, and reporting progress to the cell and other entities in the enterprise. The time required to complete an activity can be flexible, guided by the requirement to finish as early as possible. The standard computer domain covers a wide

variety of environments and operating systems, e.g., UNIX, DOS, non-RT POSIX compliant systems, and Windows NT.

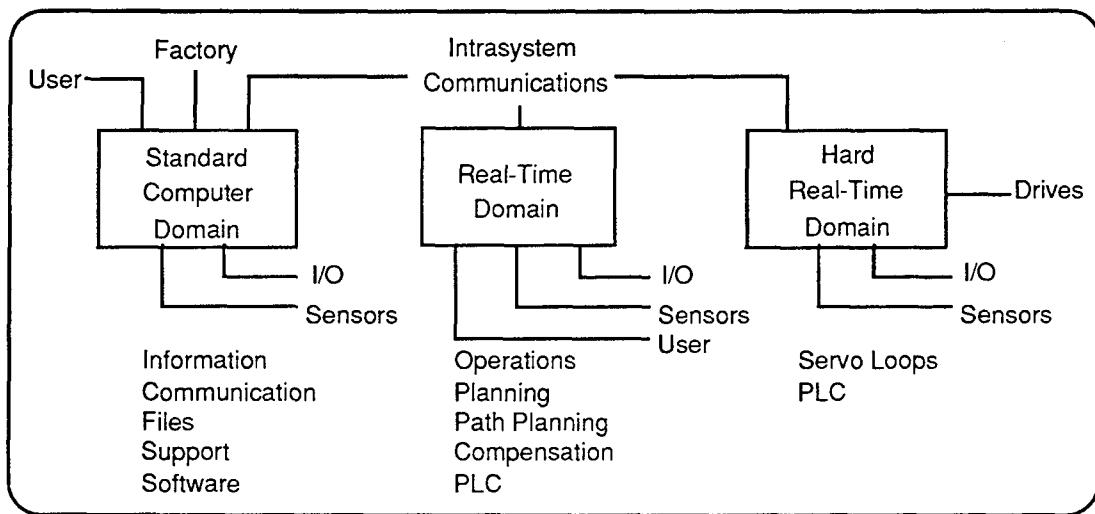


Figure 8 . NGC Timing Domains

An activity in the RT domain is driven by a time budget, but the consequences of missing a deadline are not catastrophic, and the system can recover without serious loss. Some examples of RT activities are operations planning, task planning, path planning, cutter compensation, and a PLC's actuation of the coolant valve. Hard-RT activities have mandated timing deadlines; missing one of these deadlines can have serious consequences like a damaged part that must be reworked or discarded. Hard-RT constraints apply to servo loop closures and programmable logic controller (PLC) cycles that read values from sensors into memory and write values from memory to actuators.

The need to consider the three timing domains will persist in spite of the ever-improving performance of available computing. As better computers are made, the increased throughput makes more complicated algorithms feasible within a fixed time limit. When they are deployed in advanced manufacturing applications, greater machining precision at higher speeds will be made possible. The result is better quality products at faster production rates. Thus the RT envelope will be pushed continually, and the relevance of the three timing domains will remain as more capable processors emerge.

NGC does not, at this time, directly address the issue of timing performance for resulting system implementations. In all likelihood, this will require dedicated tools that, in addition to insuring other NGC requirements are met, also evaluate the adequacy of the the overall

software/platform system with respect to performance/stability driven performance requirements. The issue of tools evolution and availability was discussed briefly in the introduction section and will be discussed later.

2.6 Open System Architecture Specification

Controller designs are constrained from two aspects: the Reference Architecture and open standards (such as POSIX) and industry conventions (such as DOS). For a specific industrial application, controller primitive components are selected from the Reference Architecture according to application-specific requirements and synthesized into an Application Architecture. The resulting Application Architecture constrains the design of the application system; that is, the system elements specifically required for manufacturing and control. The other constraining aspect supports the operation of the controller, which is based upon open and de facto standards of practice in manufacturing, controllers, computing, and data communication.

2.7 Open System Environment Overview and Profiles

The NGC is based on open standards. Open standards help to achieve a level of portability and interoperability between multi-vendor controller products that does not exist in controllers today. This “openness” of the controller facilitates the addition of new controller features and innovative technology with a relative ease that is unavailable in controllers that are closed. The benefits are two-fold: (1) the end-user has a controller that is adaptable to market dynamics and can be easily modified to incorporate the latest cost-saving technologies, and (2) third-party vendors are encouraged by the opportunities to develop new technologies and market niches for a new generation of controllers. The basic model used for describing the NGC Open Systems Environment (OSE) is very closely related to that used in the POSIX specification. As shown in Figure 9, the NGC model also relies on Application Program Interfaces (API) and External Environment Interface (EEI) definitions.

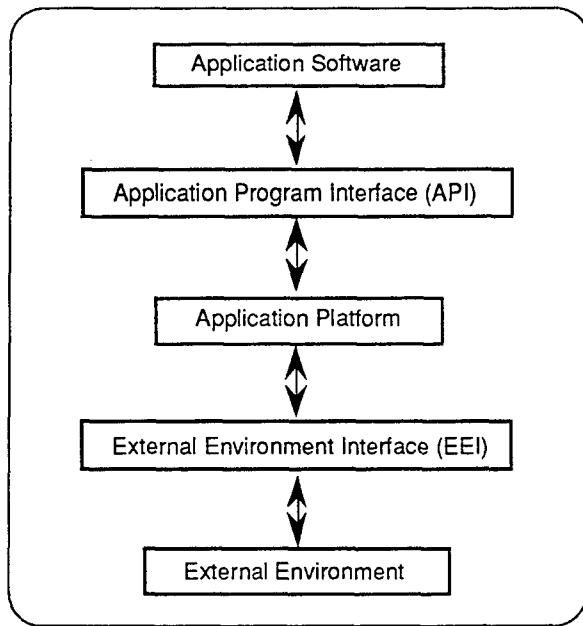


Figure 9 NGC OSE Reference Model

Central to the open theme is the NGC OSE framework. The OSE framework leverages open standards, both *de jure* and *de facto*, to specify an infrastructure for open controllers. The OSE framework embodies three general concepts: the reference model, the taxonomy, and profiles. The OSE reference model is the context for NGC open standards, the taxonomy is the logical grouping of these standards, and the profiles are selections of standards from specific groups. Rather than attempting to mandate a limited set of standards, the concept of the profile allows the vendor and the user to communicate key structural aspects of a system via a "snapshot" that, through the shorthand afforded by defined standards, allows system structure to be quickly determined. This is very similar to the common practice today of indicating whether products are "Mac" or "PC" compatible. The OSE framework is essentially the organized menu for this selection. While the possible number of permutations and combinations of profiles is overwhelming at this point, it is expected that as more vendors produce NGC systems, a smaller set of accepted profiles will emerge. Figure 10 shows a taxonomy of the standards and conventions that have been considered as possible options for NGC conformant systems.

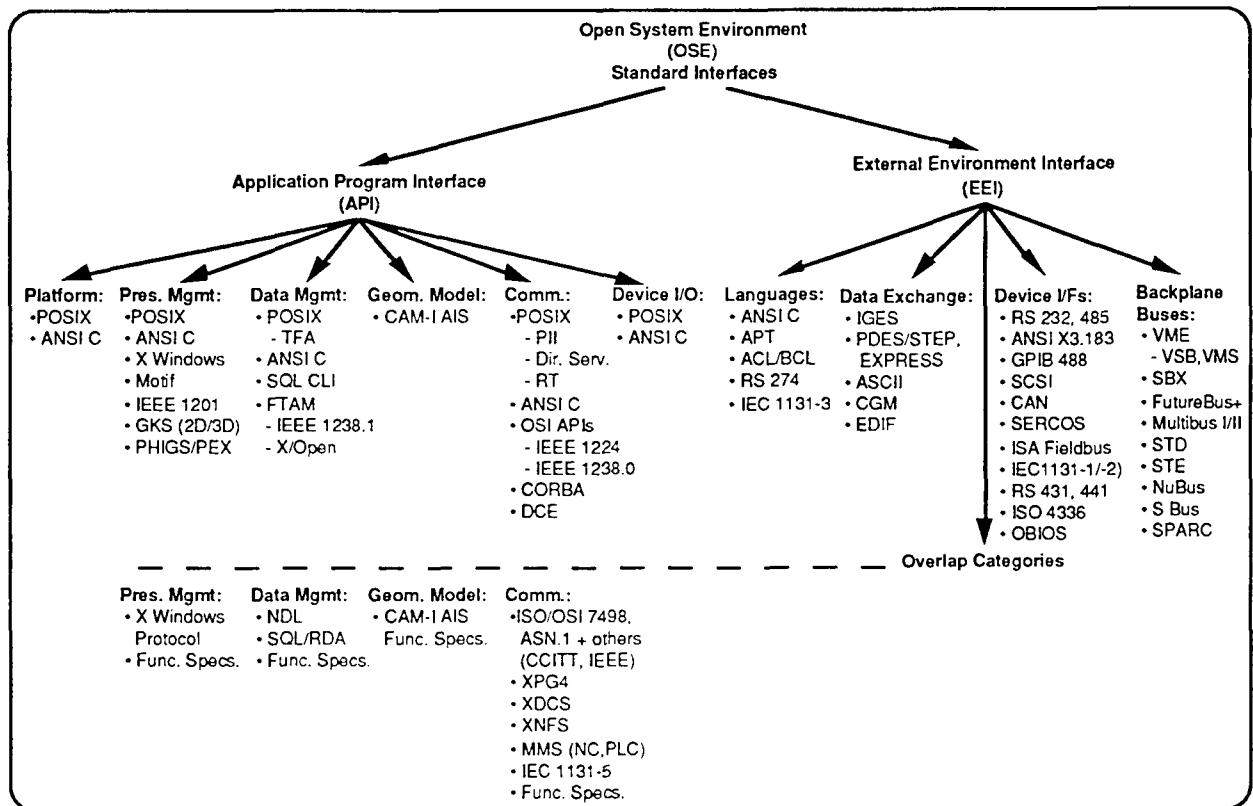


Figure 10 NGC OSE Taxonomy

Through the use of the OSE structure, a large range of implementation solutions are possible as a function of user needs and requirements. Figure 11 illustrates how the standards listed in the OSE taxonomy support, for example, a hierarchy of possible communication architectures. This reinforces the notion that NGC is not mandating a point solution with respect to hardware or software, but has the flexibility, via the underlying standards to adequately support any design regardless of how simple or how complex.

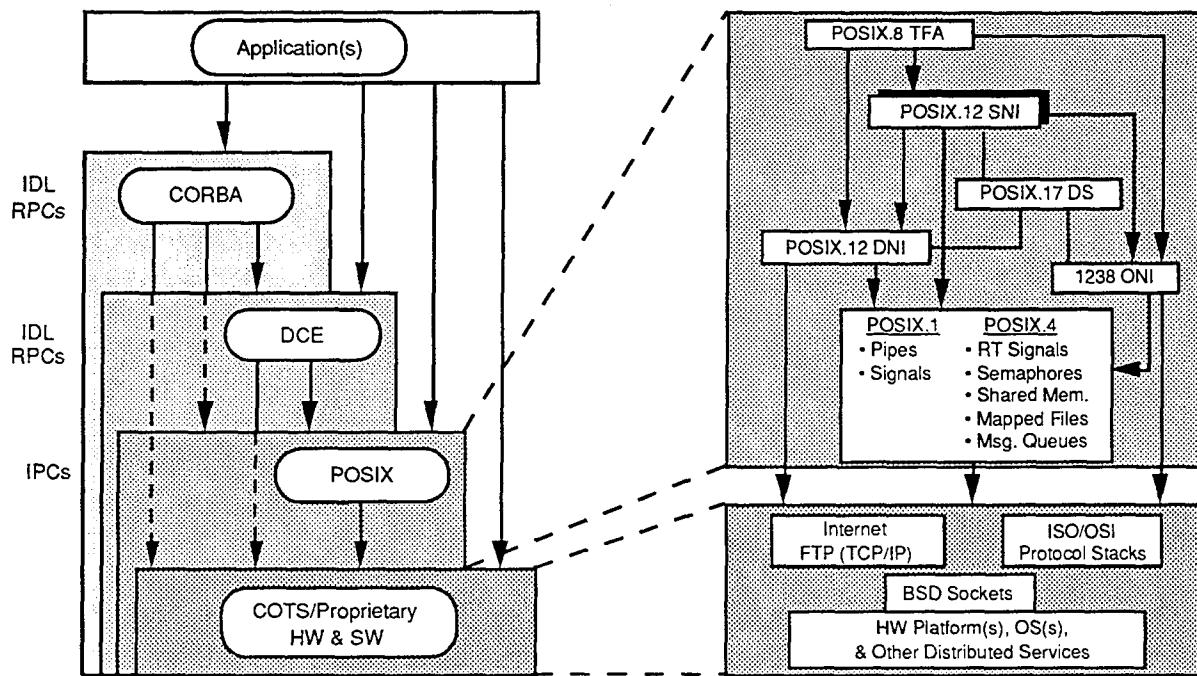


Figure 11 Communications Standards Architectural Hierarchy

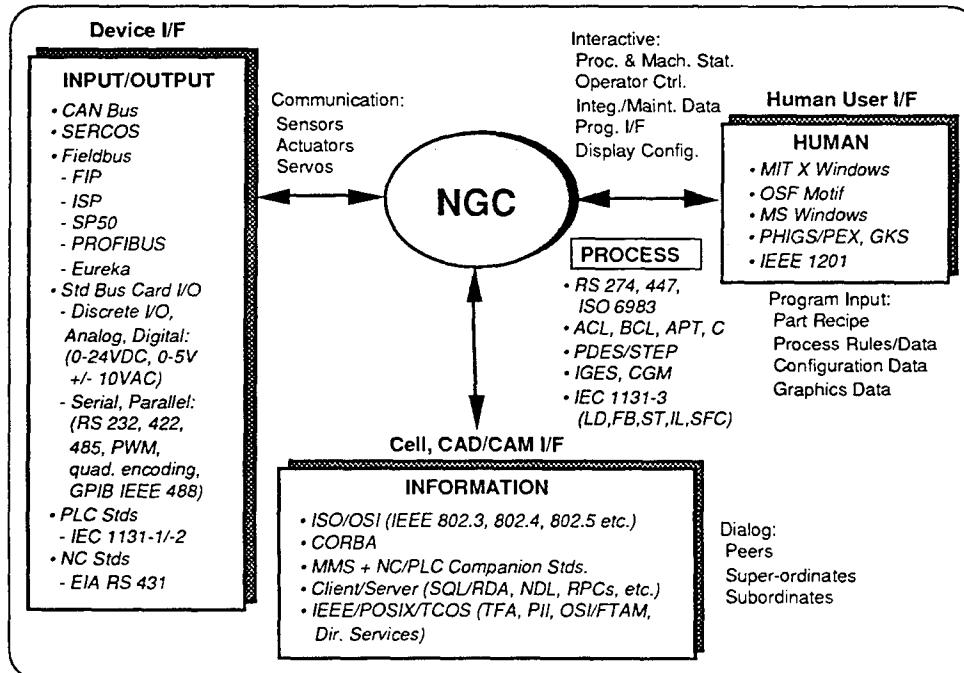


Figure 12 NGC EEI Interoperability Standards—Physical Mapping

A physical mapping of the EEI interoperability standards is shown in Figure 12. This better illustrates perhaps the most central strength of the NGC architectural structure; the ability to

main a consistent system structure while maintaining the freedom to "fine tune" specific implementation details.

Additional detail with respect to potential EEI standards is shown in Table 1. This table focuses on device interface standards.

Table 1- Device Interface EEI—Relevant Open Standards

ANSI/EIA/TIA 232-E-1991	Interface between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange
EIA 485	Standard for Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems
ANSI/IEEE 488.1-1987	Digital Interface for Programmable Instrumentation
ANSI/IEEE 488.2-1987	ICodes, Formats, Protocols, and Common Commands
ANSI X3.183-1991	Information Systems - High Performance Parallel Interface
ANSI X3.131-1986	Small Computer Systems Interface (SCSI)
ISO 9316:1989	Information processing systems - Small Computer System Interface (SCSI)
SAE J1583	Controller Area Network (CAN) An In-vehicle Serial Communications Protocol
IEC 44(secretariat)148	Serial Data Link for Real Time Communication Between Controls and Drives (SERCOS)
ISA-S50.02-1992	Fieldbus Standard for Use In Industrial Controls Systems - Part 2: Physical Layer Specification and Service Definition
ANSI/EIA 431-1992	Electrical Interface Between a Numerical Control and Machine Tools
IEC 1131-1/2 (1992)	Programmable Controllers. Part 1: General information, Part 2: Equipment requirements and tests
EIA 441-1979 (R1992)	Operator Interface Functions of Numerical Controls
ISO 4336:1981	Numerical control of machines - Specification of interface signals
OBIOS 1.15	OBIOS: Open Basic Input Output System Draft Specification - Real-Time Consortium

Because NGC allows consideration of the standards that a designer feels are relevant another mechanism from the POSIX standard has been adopted and modified to "keep track" of detailed system structure. This mechanism is the *profile*. The profile is a systematic way of representing the structure of the overall system. An example profile sheet is shown in Figure 13a.

Target Processor (Platform Only)		API Profile Suite Platform/Application				POSIX RT Profile (RT Platform Only)							
<input type="checkbox"/> CISC <input type="checkbox"/> RISC <input type="checkbox"/> DSP <input type="checkbox"/> Other Supplier/Bd./Chip: _____		Processing Requirements				<input type="checkbox"/> MINimal Embedded <input type="checkbox"/> CONtroller <input type="checkbox"/> DEDicated <input type="checkbox"/> MULTipurpose <input type="checkbox"/> Custom <input type="checkbox"/> Non-POSIX							
Application: _____ (Component/Aggregate Only)		<input type="checkbox"/> Non-RT <input type="checkbox"/> RT											
Development Language (Application Only)		Presentation Mgmt.:											
<input type="checkbox"/> Stand. C (opt. C++): <input type="checkbox"/> Diagnostics <input type="checkbox"/> Character Handling <input type="checkbox"/> Localization <input type="checkbox"/> Mathematics <input type="checkbox"/> Non-Local Jumps <input type="checkbox"/> Input/Output <input type="checkbox"/> General Utilities <input type="checkbox"/> String Handling <input type="checkbox"/> Date and Time <input type="checkbox"/> Common C (opt. C++)		Command I/F: <input type="checkbox"/> POSIX <input type="checkbox"/> FORTRAN <input type="checkbox"/> ADA <input type="checkbox"/> Non-POSIX binding <input type="checkbox"/> Pascal <input type="checkbox"/> Smalltalk <input type="checkbox"/> Lisp <input type="checkbox"/> PL/I <input type="checkbox"/> COBOL <input type="checkbox"/> Other: _____				Graphics: <input type="checkbox"/> GKS <input type="checkbox"/> 2D <input type="checkbox"/> 3D <input type="checkbox"/> Other: Prog. I/F: <input type="checkbox"/> POSIX <input type="checkbox"/> Pro. I/F: _____ <input type="checkbox"/> ANSI C <input type="checkbox"/> (Appl. Only)		Windows/GUIs <input type="checkbox"/> X Windows <input type="checkbox"/> Xlib-library <input type="checkbox"/> Xt toolkit <input type="checkbox"/> Non-Xt toolkits <input type="checkbox"/> PHIGS <input type="checkbox"/> Proprietary: <input type="checkbox"/> HOOPS <input type="checkbox"/> OpenGL <input type="checkbox"/> GDI <input type="checkbox"/> Quickdraw <input type="checkbox"/> Other: _____ <input type="checkbox"/> IEEE 1201		<input type="checkbox"/> Non-X Win/GUIs: <input type="checkbox"/> DOS Text/Graphics <input type="checkbox"/> Windows <input type="checkbox"/> OS/2 <input type="checkbox"/> Windows NT (Win32) <input type="checkbox"/> Mac Windows <input type="checkbox"/> Other: _____			
Platform:		Data Mgmt.:				Comm.:		Geom. Model.:					
<input type="checkbox"/> POSIX <input type="checkbox"/> POSIX.1 <input type="checkbox"/> POSIX.4 <input type="checkbox"/> POSIX.4a <input type="checkbox"/> POSIX.4b <input type="checkbox"/> ANSI C (Appl. Only)		<input type="checkbox"/> Non-POSIX <input type="checkbox"/> type: _____ <input type="checkbox"/> UNIX <input type="checkbox"/> DOS/Windows <input type="checkbox"/> OS/2 <input type="checkbox"/> NT <input type="checkbox"/> Mac OS <input type="checkbox"/> Other: _____				<input type="checkbox"/> POSIX <input type="checkbox"/> POSIX.1 <input type="checkbox"/> POSIX.4 <input type="checkbox"/> POSIX.4a <input type="checkbox"/> ANSI C <input type="checkbox"/> (Appl. Only) <input type="checkbox"/> SQL CLI <input type="checkbox"/> FTAM <input type="checkbox"/> IEEE 1238.1 <input type="checkbox"/> X/Open		<input type="checkbox"/> POSIX.8 (TFA) <input type="checkbox"/> Core TFA <input type="checkbox"/> Full TFA <input type="checkbox"/> Profiled: (RFS, NFS, FTAM, PC/DOS, other) <input type="checkbox"/> Other: _____		<input type="checkbox"/> POSIX.17 <input type="checkbox"/> POSIX.21 <input type="checkbox"/> ANSI C <input type="checkbox"/> OSI APIs <input type="checkbox"/> -IEEE 1224 <input type="checkbox"/> -IEEE 1238.0 <input type="checkbox"/> -X/Open <input type="checkbox"/> CORBA <input type="checkbox"/> DCE <input type="checkbox"/> POSIX.12 <input type="checkbox"/> Other		<input type="checkbox"/> CAM-I AIS	
								Device I/O:					
								<input type="checkbox"/> POSIX <input type="checkbox"/> POSIX.1 <input type="checkbox"/> POSIX.4 <input type="checkbox"/> POSIX.4b <input type="checkbox"/> ANSI C <input type="checkbox"/> Other: _____					

Figure 13 a API Profile Suite

This illustrates the fundamental attributes of the profile. It allows the vendor to specify which standards or conventions are adhered to here as well as specific implementations within the identified standard. To reiterate it is, on a more complex level, equivalent to specifying "Mac" or "PC".

Figure 13b shows the current concept for a template structure that identifies all of the *possible* profile categories that can be used for a full system description. The word possible is highlighted because it is highly likely that most systems can be adequately described by a set of profiles much smaller than the complete set. It is reasonable to assume that, as the profile convention is adopted on a wider basis by the community, the profiling process will become much more efficient and will be tailored to specific community segments. For example, a reduced set of profiles might be used for specification of commercial products where only top level information is needed with respect to basic system compatibility. On the other hand, developer's will probably require a richer set of profiles for design and development activities.

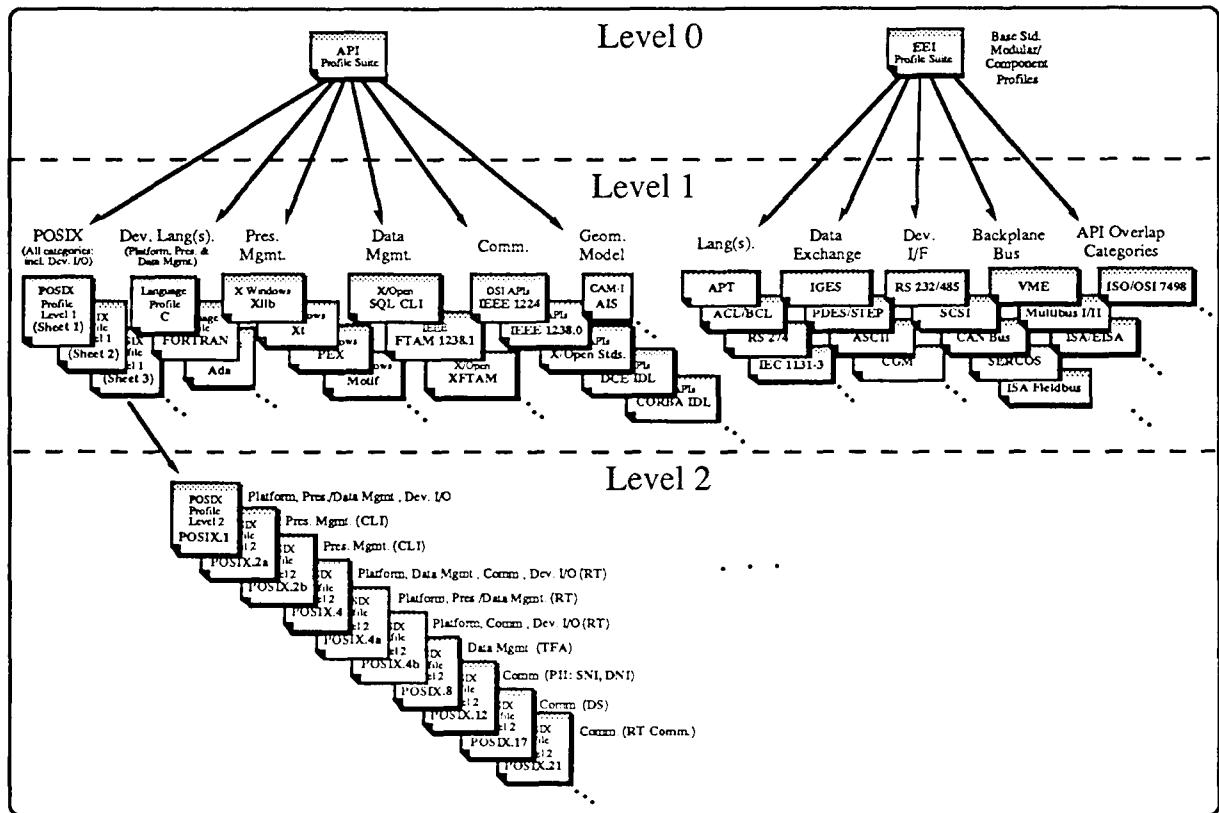


Figure 13b NGC Profiles—Template Structure

2.8 Implementation Issues

Previous sections have described the overall structure and terminology associated with the architectural concepts that serve as the foundation for NGC. This section illustrates some of the concepts presented in the context of a controller for a machining center.

An example of an application architecture for a machining center is shown in Figure 14. This demonstrates the way in which aggregate components are synthesized from primitive components. (Primitive components are shown as PC: xx in each of the aggregate component blocks.) This figure better illustrates that the component based approach for defining systems is really nothing more than a rigorous and systematic way of generating the block diagrams so familiar in all forms of engineering.

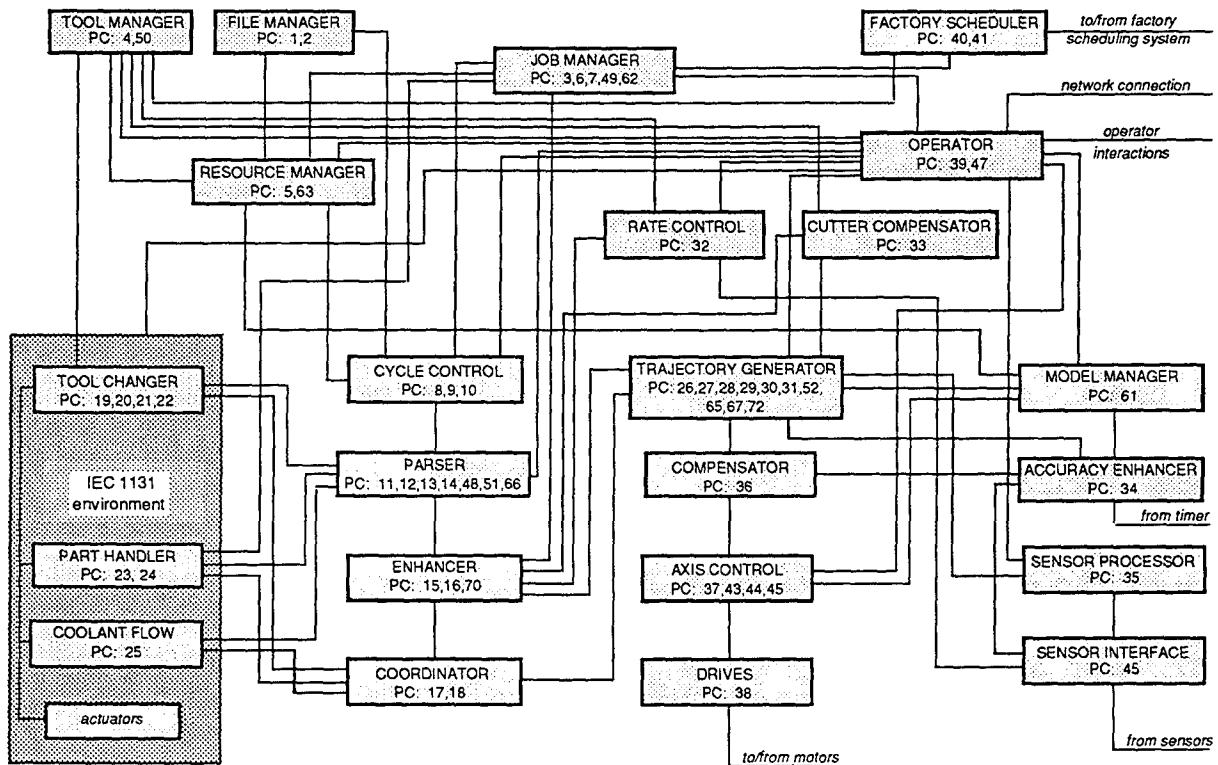


Figure 14 . Example of an Application Architecture for a Machining Center

Figure 15a and 15b show an example of two different ways that an application architecture could be partitioned in the implementation process. Figure 15a shows a system that consists of a non-realtime and a dedicated realtime platform. Another option, however, would be that as shown in Figure 15b. In this case the non-realtime and realtime platforms are combined with a hard realtime platform, such as one of widely available motion control boards with a dedicated DSP processor. These examples are not meant to advocate a particular problem solution, but simply to illustrate the overall process involved in moving from Application Architecture to Application System.

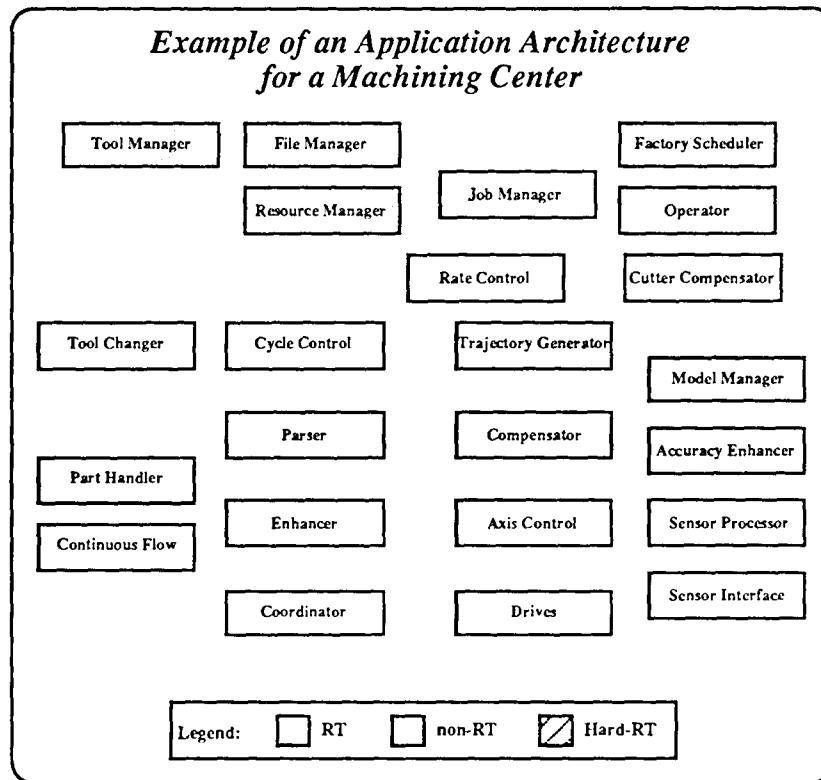


Figure 15a Platform Partitioning, Example 1

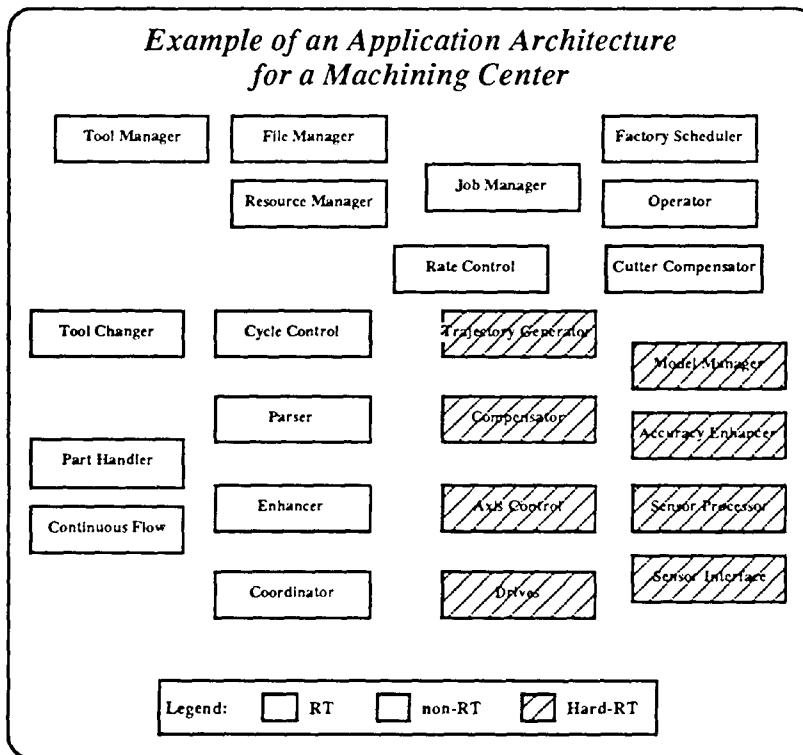


Figure 15b Platform Partitioning, Example 2

Finally, Figure 16 illustrates two possible implementations corresponding to the platform partitioning shown in Figure 15a. Suitably documented, both of the systems shown are NGC conformant and both can result in fully open systems.

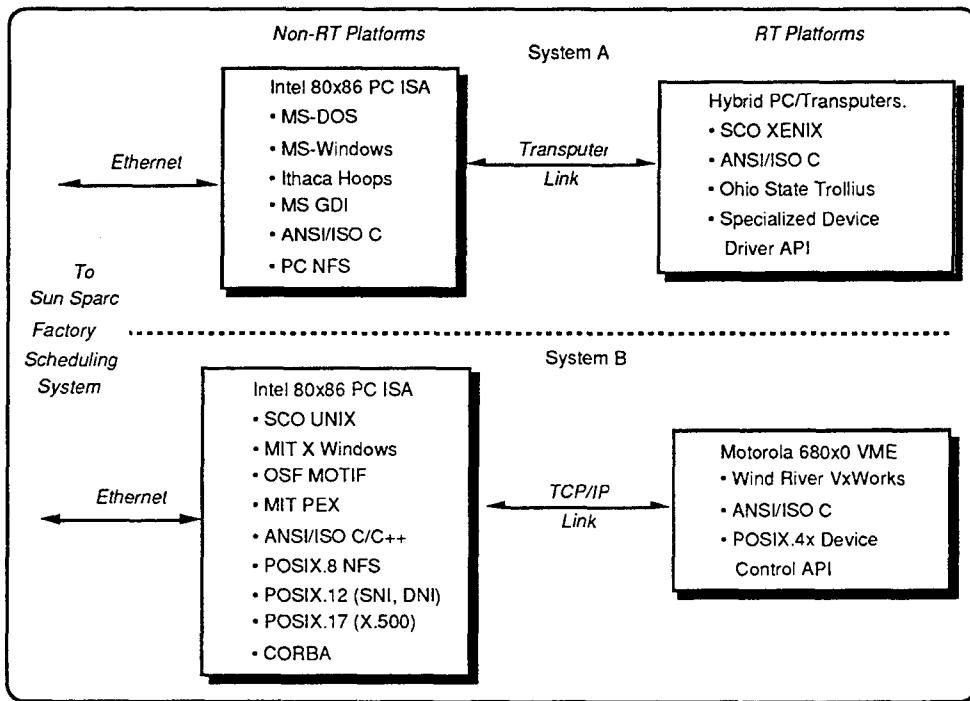


Figure 16 Platform Implementation Examples

2.9 NGC Tools and Libraries

The intense competitive pressures and risks associated with development and subsequent production of industry-fielded, open architecture, machine controllers necessitates the wise and prudent use of development and validation tools, such as knowledge-based tools, libraries, and object-modeling tools. Tools should address system design and integration support; configuration management; event timing; profile mapping and applications supported; and the taxonomy and hierarchy of events, applications, standards, agents, and components. Metrics and measurement methodologies need to be defined, developed, and tested for validity and significance and then be evaluated for the benefits, capabilities, and features of a specific implementation. Performance testing should consider the related open system specifications, data exchange and information handling protocols, external and internal interfaces, and the explicit and implicit features of openness, i.e., portability, interoperability, scalability, interchangeability, commonality of components, etc. The

development and validation tools, and documentation of lessons learned, should be mapped to the capabilities and features of an open architecture machine controller.

Many software tools that would fulfill requirements for the NGC tool suite are either available on the market today or are being developed in other ongoing programs. The ARPA Domain Specific Software Architecture (DSSA) program is in the process of establishing many of the fundamental tools that would be required in the NGC requirements definition, design, and validation and verification activities. As NGC continues to evolve and mature, it will be essential to find mechanisms for capturing the tools legacy that exists and effectively integrating it into the NGC structure.

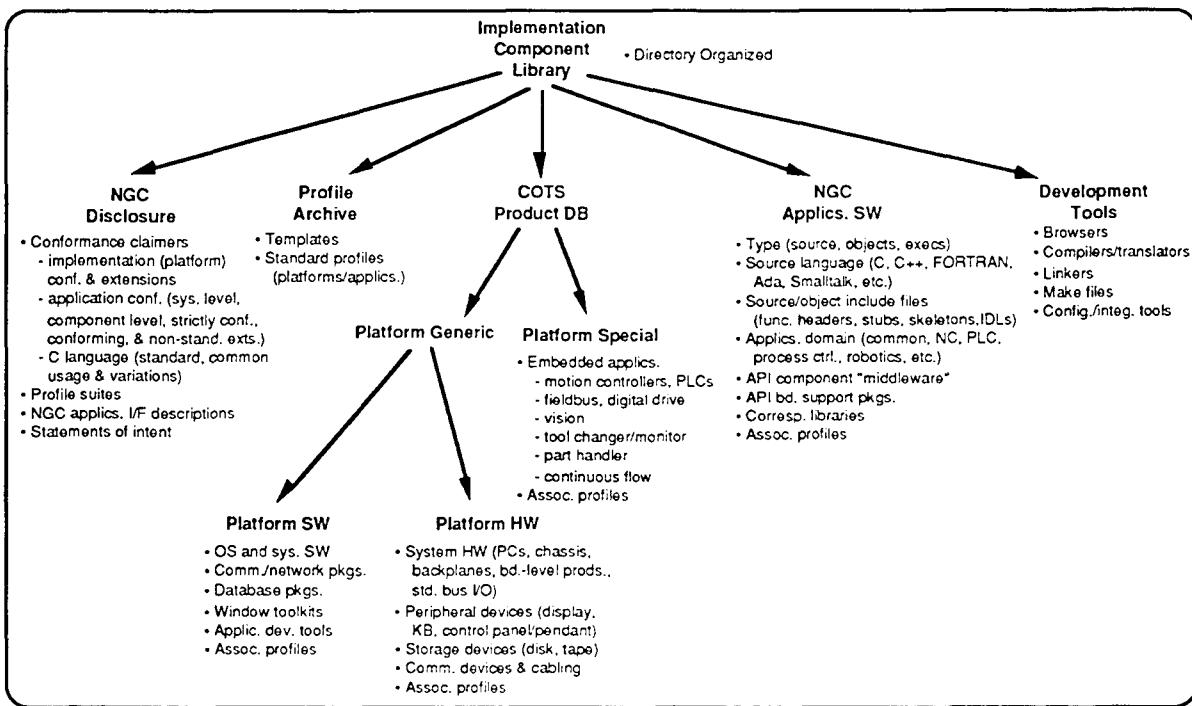


Figure 17 Implementation Component Library & Tools—Structure & Contents

2.10 Conformance Overview

NGC conformance is determined by adherence to a specific profile of standards. The NGC SOSAS does not attempt to specify a standard set of profiles. The market will drive out the set of profiles that are most practical to both the controller developer and end-user.

A NGC-conformant product must include a “NGC Disclosure Statement” that specifies the profile, component interfaces, and other conformance claimers where applicable.

Profiling offers the controller developer options for product conformance at a variety of levels to satisfy user requirements at a competitive cost, with varying degrees of openness.

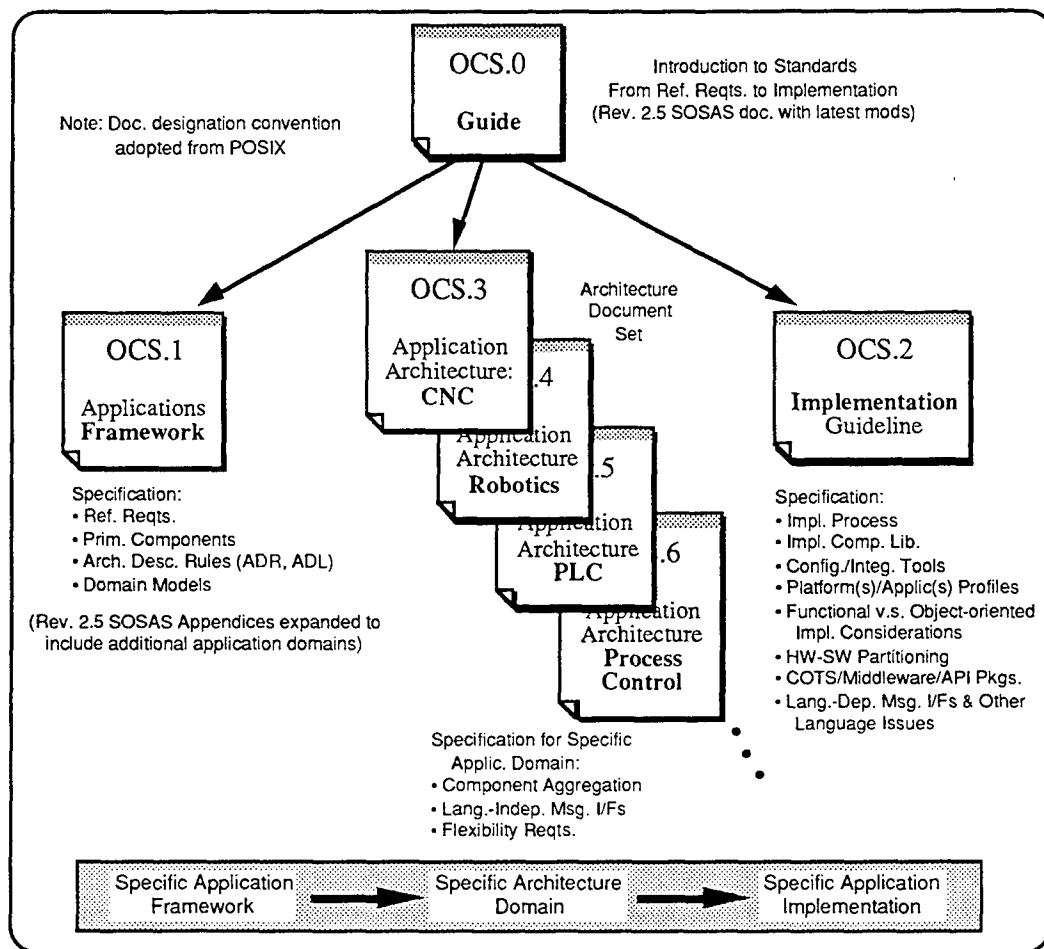


Figure 18 NGC Document Vision: Open Controller Standards

2.11 Growth and Evolution

This document should be viewed as the initial document in what will be a growing set of companion documents that govern the full NGC development process. This initial SOSAS document should be viewed as the first in a set of documents with a structure similar to that used by IEEE to document the POSIX standard. The structure of this family of documents is shown in Figure 18. Like the POSIX standard, there is an overall document that describes basic system structure and philosophy. Other documents (the .1 and .2 documents) address basic issues associated with system framework and implementation. Finally, the architecture

document set deals with domain specific considerations. As shown, it is presumed that this would initially include CNC, robotics, process control and PLC applications.

The issue of the criticality of tools to a mature NGC process was discussed in an earlier section. In addition to tools, success of NGC also hinges on the emergence of libraries (or repositories) of effective and reliable implementation components that allow the system developer to quickly satisfy both functionality and platform requirements in the implementation process. As in the case of tools, other ongoing programs have already developed much of the foundation for these libraries.

3.0 SUMMARY AND RECOMMENDATIONS

3.1 Summary

This document has described a structure for the development of open architecture systems for advanced manufacturing applications. This structure accommodates both a wide range of software applications while at the same time permitting the use of a variety of potential platform solutions. The admissible platform solutions allow for a wide range of variability with respect to processor selection, buss selection, operating systems, and device interface and communication devices and standards.

The process of NGC system development has been formalized with respect to development of the overall system structure that satisfies system requirements through the use of a component based approach that allows the design process to begin at a level of abstraction that focuses on responsibilities and information flow within the system. This flow from the selection of primitive components from a reference architecture to the construction of an application architecture was chosen because of it's consistency with what is becoming common practice in the software community. This follows the same basic practices set forth in the Advanced Research Project Agency (ARPA) Domain Specific Software Architecture (DSSA) program.

The equally challenging issue of the system platform has been addressed through the concept of profiles, a concept adopted from the IEEE Portable Operating System Interface (POSIX) standard. The profile is essentially a means of declaring a specific implementation

of a system by selecting standards and options from a large set of possible choices. The strategy of profiles was chosen as an alternative to selecting a small set of standards and conventions from a very large set and attempting to force this on the overall community. As stated in the introduction of this document, it will be the inevitable market forces that will result in a narrowing of design options for NGC systems and the emergence of a small set of standard implementations such as the PC and the Macintosh in the personal computer arena. Realistically, any attempt to guess which set of standards will win with respect to processors, busses, operating systems, device interfaces, etc. is futile. The profiling concept necessarily leads to the possibility of an extremely large number of NGC systems, at least initially. In some ways this can be viewed as no different from the current situation. There is, however, a crucial difference. Profiling allows users to determine, with little effort, the degree of openness in the system that is being purchased. By declaring the fundamental structure of the system, the path is opened for other vendors to produce system elements that can easily be integrated into the resulting standardized systems. The ultimate benefit is to the user who, using the system profile, can draw on a wide range of related products for system expansion and improvement.

The profile system that has been developed addresses both the Application Program Interface (API) as well as the External Environment Interface (EEI). (Again, this is similar to POSIX). Taken together, the application architecture and the profiling structure form a firm foundation for NGC system development; they do not, however, represent a complete methodology that would support immediate development of a national,commercial NGC product base. There are two elements that are missing from the overall system that would facilitate the adoption of the NGC approach; *fully populated and widely accessible implementation component libraries* and a family of *tools for system design, integration, and validation*. Unfortunately, the difficulty lies in developing the initial libraries and tool set. Once initiated, it seems clear that they will themselves become important commercial products.

At the implementation component level abstraction is finally lost and issues of system interoperability are directly addressed. The implementation library, (or repository in DSSA terms), consists of elements of software that are actually linked and compiled into the final system. From the standpoint of the end user the truest manifestation of NGC is in the availability of implementation components, not in the abstract but necessary process that results in these components. It is not an over simplification to say that the implementation components will be the "currency" of the evolutionary NGC system.

Even with the availability of libraries of implementation components, computer based tools will be essential to the propagation and success of this technology. The tool set is what will be responsible for institutionalizing the full NGC process because it will provide a quantum leap in the way systems are developed and modified in the same way Windows and Macintosh interfaces represented a quantum leap with respect to humans interfacing with computers.

3.2 Synergistic Activities

There are a large number of ongoing activities that have the potential for filling in the remaining pieces of the NGC structure. These activities include controller development projects as well as more generalized realtime, distributed architecture projects. The controller projects provide much of the necessary detail needed to arrive at a complete and useful NGC structure. Specifically, they provide a set of primitive components, requirements definition, application architectures, but most importantly, specific messaging structures and an initial set of implementation components. It is also important to note that the identified controller projects are not limited to strictly machine tool control architectures. In addition to general machine tool applications, there are ongoing projects that also address inspection (coordinate measurement) and robotics (both autonomous and teleoperated.) Similarly, a number of ongoing realtime, distributed control projects are focusing on many of the "domain independent" communication issues that are important to NGC as well as the tool structure for both building and maintaining NGC-like systems.

The controller projects that could directly contribute to completing the overall structure of the NGC system include the NIST Enhanced Machine Controller, (EMC), the Unified Telerobotic Architecture Project, (UTAP), sponsored by the Aircraft Directorate at the San Antonio Air Logistics Center, (SA-ALC), with participation by NIST and JPL, the Air Force ManTech Title III activity, the Department of Energy Technologies Enabling Agile Manufacturing, (TEAM), as well as others. Even the European version of NGC, Open System Architecture for Controls within Automation Systems, (OSACA), is worthy of further study with respect to what it can offer a long term NGC activity. All of the programs described above offer elements that can be directly incorporated into the evolving NGC structure that was shown in Figure 18. The NGC structure benefits from the standpoint of additional depth and completeness. Each of the individual programs also benefits from the ability to capture program legacy in the consistent framework of NGC.

Similarly, there are a number of projects that are addressing many of the domain independent and tools issues that are crucial to an effective and complete NGC system. The ARPA DSSA program is currently felt to be the most important among these programs. This program is working many of the issues essential to the advanced manufacturing domain. Issues that are currently at the forefront of this project include requirements, system structural development, testing, and validation of systems. Most importantly, DSSA is focusing on a consistent set of tools that will make open, reusable systems a reality for all realtime control system applications. Even programs seemingly unrelated to NGC and advanced manufacturing such as the Reuseable Software Architecture for Spacecraft, (RSAS), sponsored by the Air Force Phillips Lab, are in the process of working issues and developing tools that are directly relevant to NGC. Finally, the National Information Infrastructure Program (NIIP) is of direct interest to the NGC development process. This program has the goal of establishing and unifying many of the key information transfer standards necessary for NGC.

3.3 Recommendation

An effective overall structure has been developed for the NGC vision for advanced manufacturing systems. This structure facilitates the development of open, interoperable controllers for all advanced manufacturing operations. At the same time, the structure allows for an unprecedented growth in the vendor base that can field products relevant to this area. It is not an overstatement to say that this structure *could* serve as the foundation for a revolution in advanced manufacturing systems similar to that which has occurred in personal and workstation computing over the past ten years.

To make the NGC vision a reality, it will be necessary to continue the development of the overall system structure. This is not an overly daunting task because, as pointed out in the earlier parts of this section, the activity should focus on capture of technology from ongoing activities rather than additional development. It is safe to say that all the major elements required to complete the NGC system either currently exist, (e.g. NIST EMC), or are in the process of being developed (e.g. ARPA DSSA, Title III, TEAM, NIIP). The integration of these available elements is not an especially complex task, but it does require that some type of central repository site be identified and solid interfaces with these programs be developed.